



GStreamer for Mobile Platforms: Android and iOS

GStreamer Conference 2013, Edinburgh

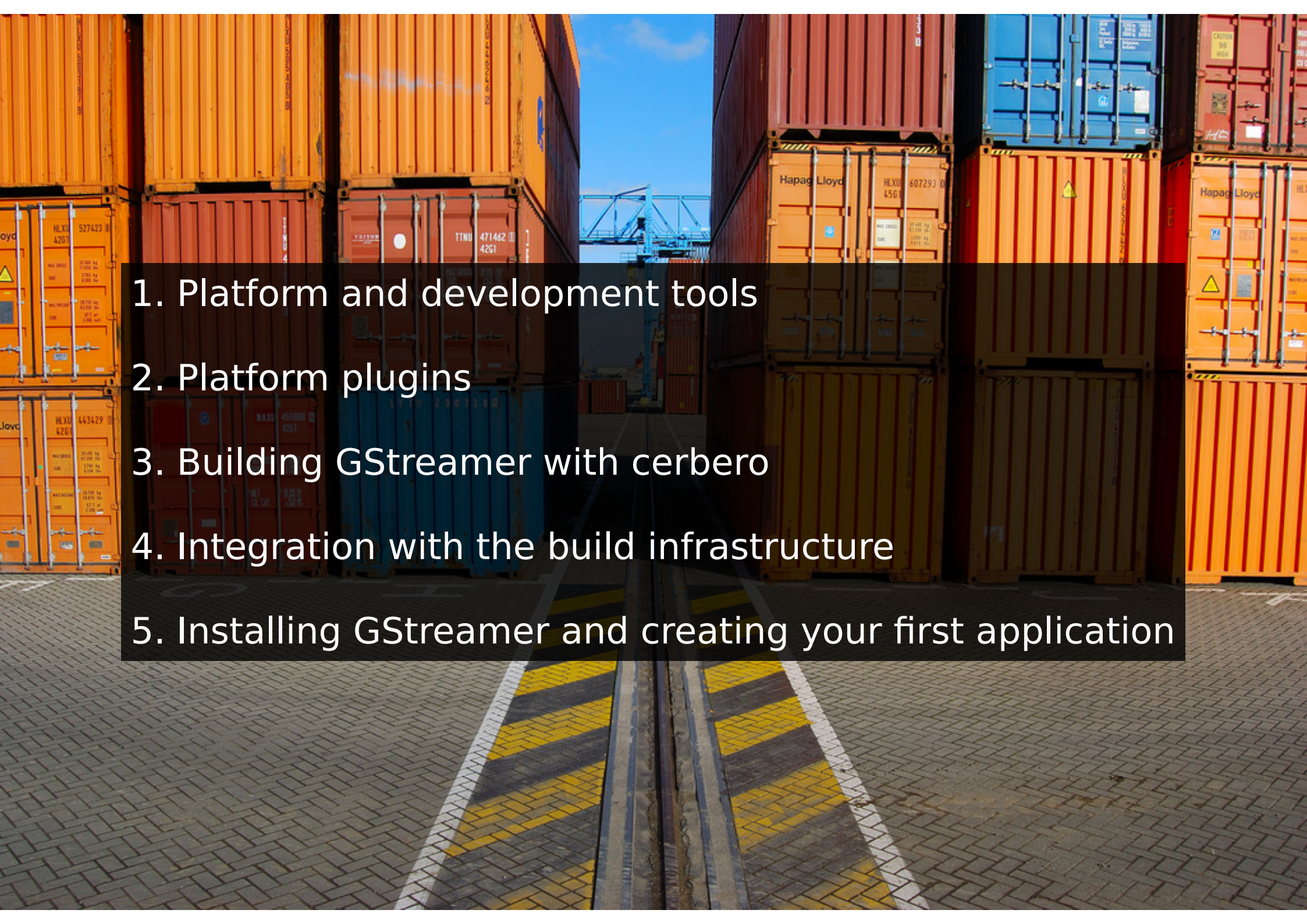
Andoni Morales Alastruey



Who am I ?

Andoni Morales Alastruey amorales@fluendo.com

- Working for Fluendo
- GStreamer contributor since 2008
- LongoMatch, a sports video analysis software using GStreamer (www.longomatch.org)

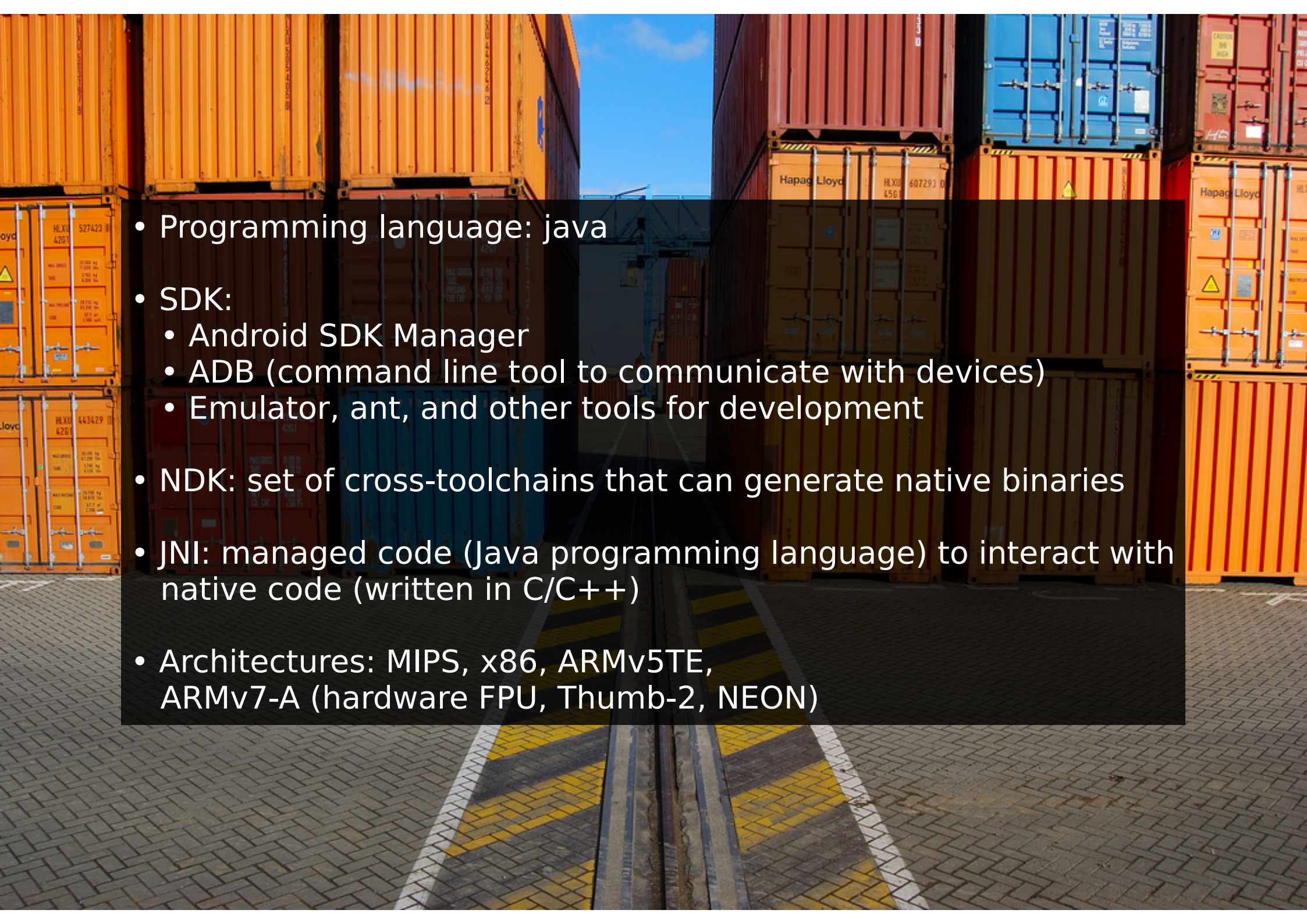
- 
1. Platform and development tools
 2. Platform plugins
 3. Building GStreamer with cerbero
 4. Integration with the build infrastructure
 5. Installing GStreamer and creating your first application

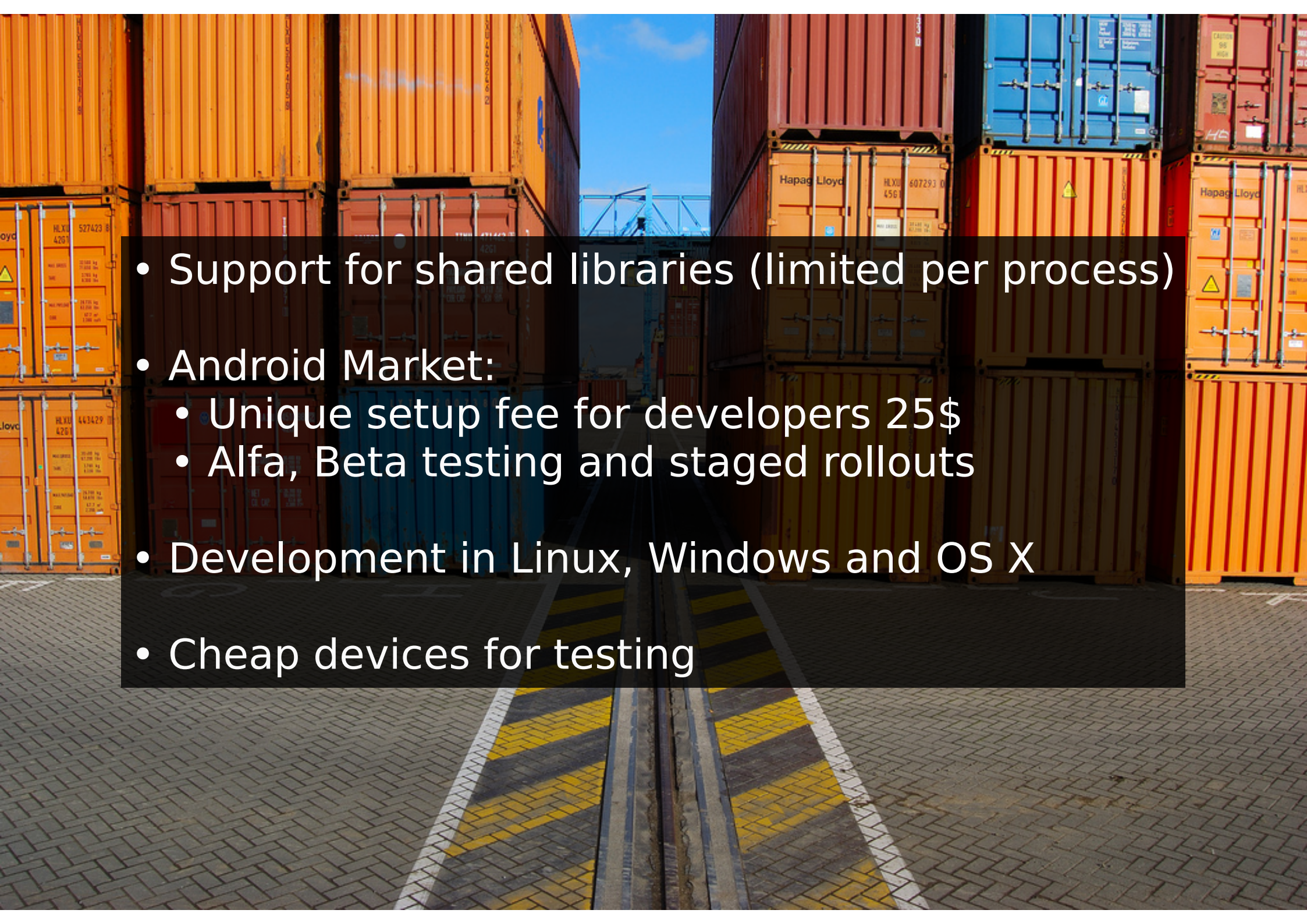


1. Platforms and the development tools



Android

- 
- Programming language: java
 - SDK:
 - Android SDK Manager
 - ADB (command line tool to communicate with devices)
 - Emulator, ant, and other tools for development
 - NDK: set of cross-toolchains that can generate native binaries
 - JNI: managed code (Java programming language) to interact with native code (written in C/C++)
 - Architectures: MIPS, x86, ARMv5TE, ARMv7-A (hardware FPU, Thumb-2, NEON)

- 
- Support for shared libraries (limited per process)
 - Android Market:
 - Unique setup fee for developers 25\$
 - Alfa, Beta testing and staged rollouts
 - Development in Linux, Windows and OS X
 - Cheap devices for testing

The NDK (Native Development Kit)

- Toolchains (compiler, linker, etc...) for the different target architectures
- ndk-build: build system to compile native code and embed it the .apk
- ndk-gdb: remote GDB debugger
- Provide a limited API:
 - libc (C library) headers
 - libm (math library) headers
 - JNI interface headers
 - libz (Zlib compression) headers
 - liblog (Android logging) header
 - OpenGL ES 1.1 and OpenGL ES 2.0 headers
 - libjnigraphics (Pixel buffer access) header
 - A Minimal set of headers for C++ support
 - OpenSL ES native audio libraries
 - Android native application APIS



Android Multimedia Stack

- OpenGL ES 2.0 + EGL (video sink)
- OpenSL ES (audio sink)
- `android.media.MediaCodec`
(video/audio decoder/encoder)
- `android.hardware.Camera` (video source)



Android Multimedia Stack

- Good multimedia API for playback and capture and rendering
- Support for most common audio and video formats
- Software and hardware decoders/encoders
- And a few streaming protocols (RTSP, HTTP, HLS)

iOS

- Programming language: Objective C
- We can use the GStreamer C API directly
- XCode:
 - Cross toolchain
 - Frameworks
- Architectures:
 - Emulator: x86
 - Devices: ARMv6 and ARMv7
- Static linking

ios

- Some parts of the API are not available in the emulator
- Applications sandboxing is insane
- App Store:
 - 100\$ per developer... per year!!!
 - You can't test your code in a real device without a developer account :(
 - Deploying apps to the App store is a long process, make sure you have everything tested.
- Development in OS X only
- Devices are not cheap
- Their business model is clear: \$\$\$\$\$\$

The background of the slide is a photograph of a shipping yard. It shows several rows of stacked shipping containers in various colors, including orange, red, and blue. The containers are arranged in a way that creates a sense of depth, with a path leading towards the horizon. The sky is clear and blue. The overall scene is industrial and brightly lit.

XCode

- IDE: similar development environment to OS X
- Cross toolchain for the emulator and the device
- iOS Frameworks
- We can use the GStreamer C API



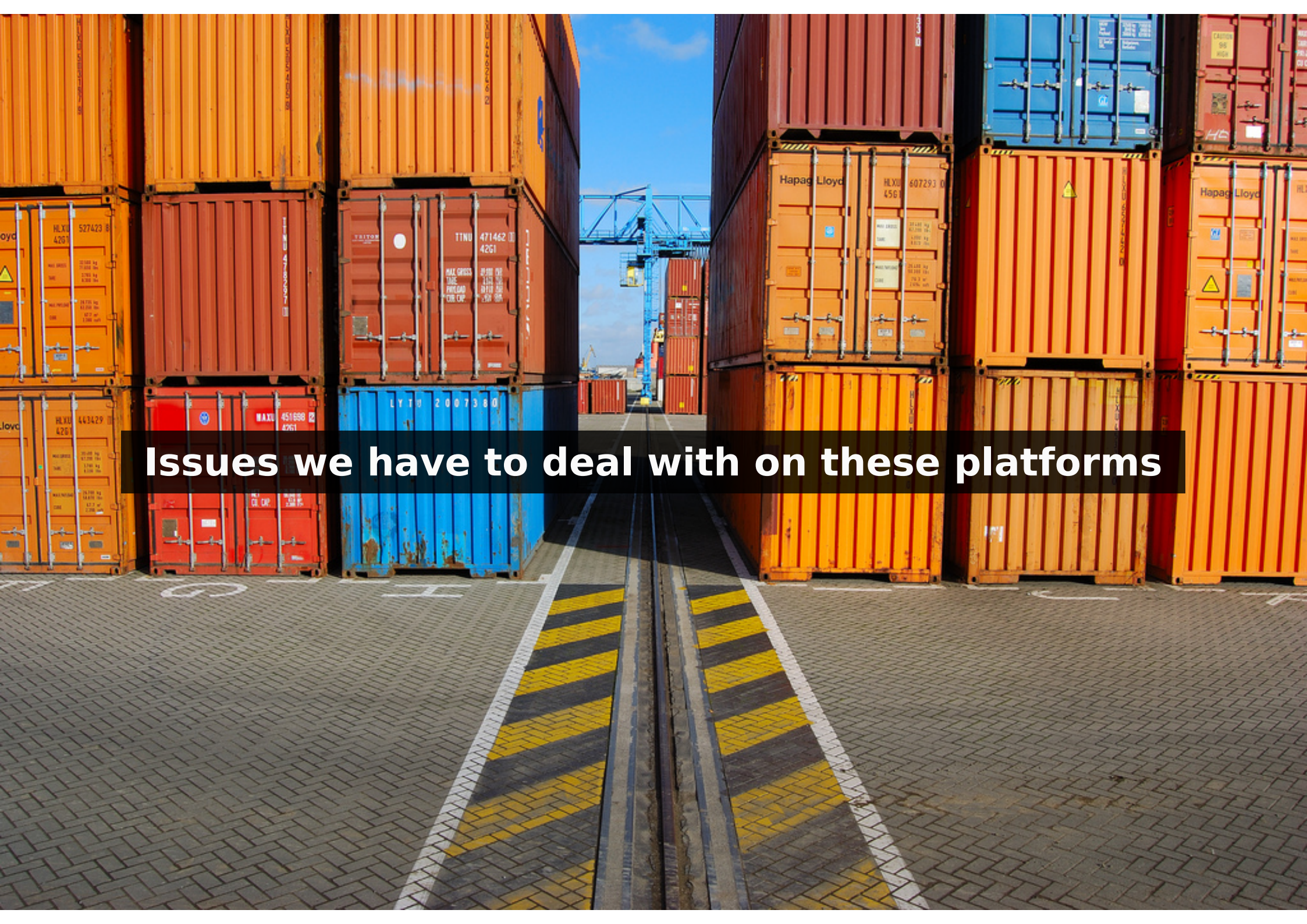
iOS Multimedia Stack

- OpenGL ES 2.0 + EAGL (video sink)
- CoreAudio Remote I/O Audio Unit (audio sink)
- No API for hardware decoders/encoders!
(VideoToolBox is a private Framework for iOS)
- AVFoundation (video source and URI decoder)

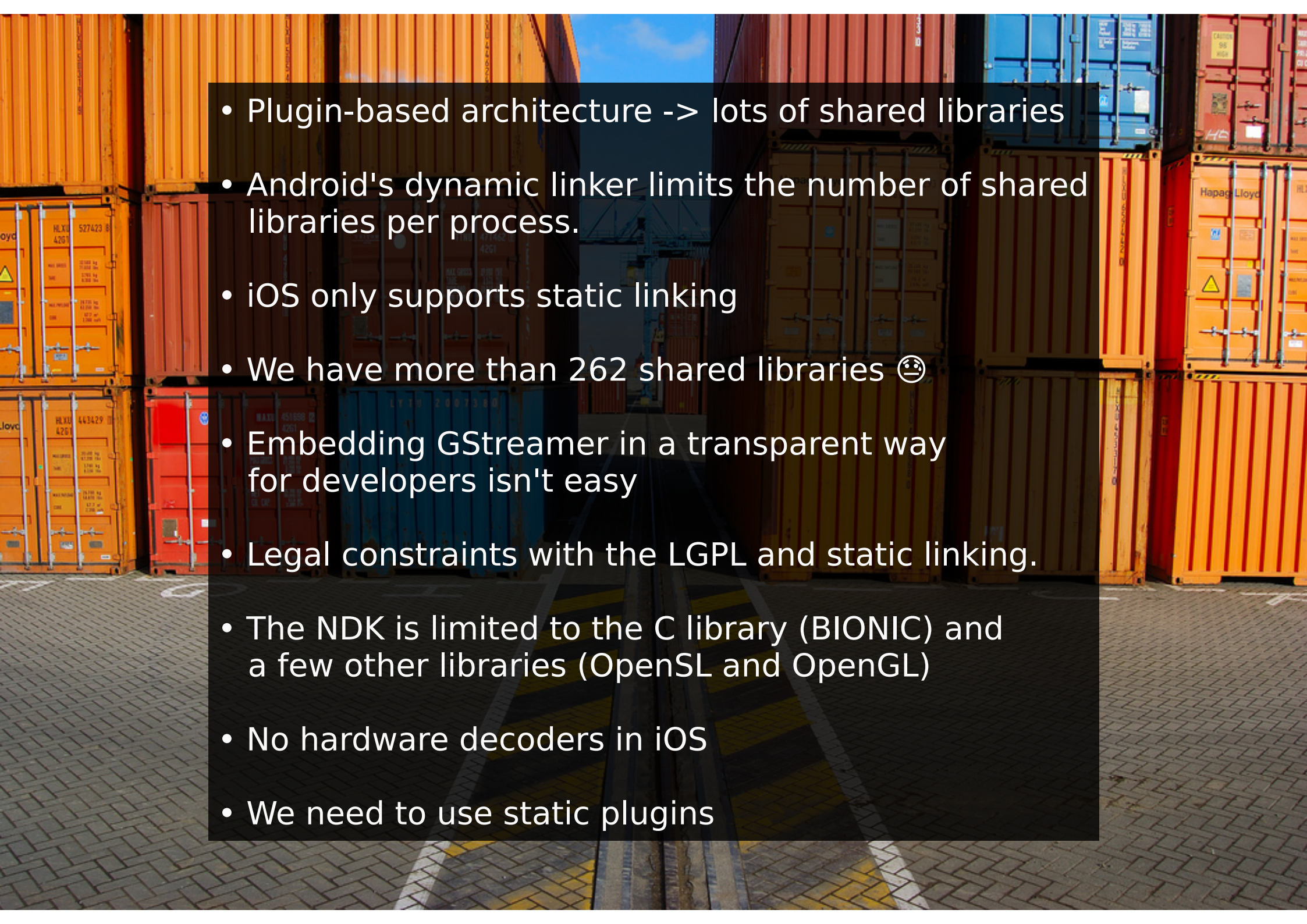


iOS Multimedia Stack

- Good API for source and sink elements
- Hardware decoding is limited to file and http uri's in a supported container
- Only a few format supported



Issues we have to deal with on these platforms

- 
- Plugin-based architecture -> lots of shared libraries
 - Android's dynamic linker limits the number of shared libraries per process.
 - iOS only supports static linking
 - We have more than 262 shared libraries 😞
 - Embedding GStreamer in a transparent way for developers isn't easy
 - Legal constraints with the LGPL and static linking.
 - The NDK is limited to the C library (BIONIC) and a few other libraries (OpenSL and OpenGL)
 - No hardware decoders in iOS
 - We need to use static plugins



2. Platform plugins



OpenGL ES + EGL/EAGL Video Sink


egllessink gst-plugins-gl

OpenGL ES + EGL/EAGL Video Sink

- Available for Android and iOS
- Public and native API
- Supports hardware accelerated colorspace conversion, scaling
- Usable on all Android (since 2.3) and iOS devices
- Works like any other GStreamer video sink
- Requires the application to provide a surface using the GstVideoOverlay interface
- Can provide a Surface + SurfaceTexture for direct rendering in Android with amcvideodec



Android



OpenSL ES Audio Sink/Source

openslessink openslessrc



OpenSL ES Audio Sink/Source

- OpenSL ES only public native API for audio on Android
- Very limited implementation available on Android
- Usable on all Android devices
- Uses Android-specific API extensions



android.media.MediaCodec Wrapper
amvvideodec amcaudiodec



Media Codec API

- Be able to use device's codecs
- Uses Java API via JNI
- Java/JNI not performance problem
- Usable on all Android devices (Jelly Bean 4.1)
- Bad colorspace definitions (vendor-specific colorspace):
`QOMX_COLOR_FormatYUV420PackedSemiPlanar64x32Tile2m8ka`
- Supports direct rendering with GL sinks

Media Codec API

- Implemented: audio/video decoders
- Encoders: patch in bugzilla
- 1080p h264 easily possible, impossible in software
- Supported video codecs:
h264/AVC, MPEG4, h263, MPEG2 and VP8
- Supported audio codecs:
AAC, MP3, AMR-NB/WB, A-Law, μ -Law,
Vorbis and FLAC



androidcamerasrc?

- No source element for now
- Can be easily implemented with the Camera Java API
- Easy to implement with appsrc for now :)



ios



AVFoundation video source
avfvideosrc

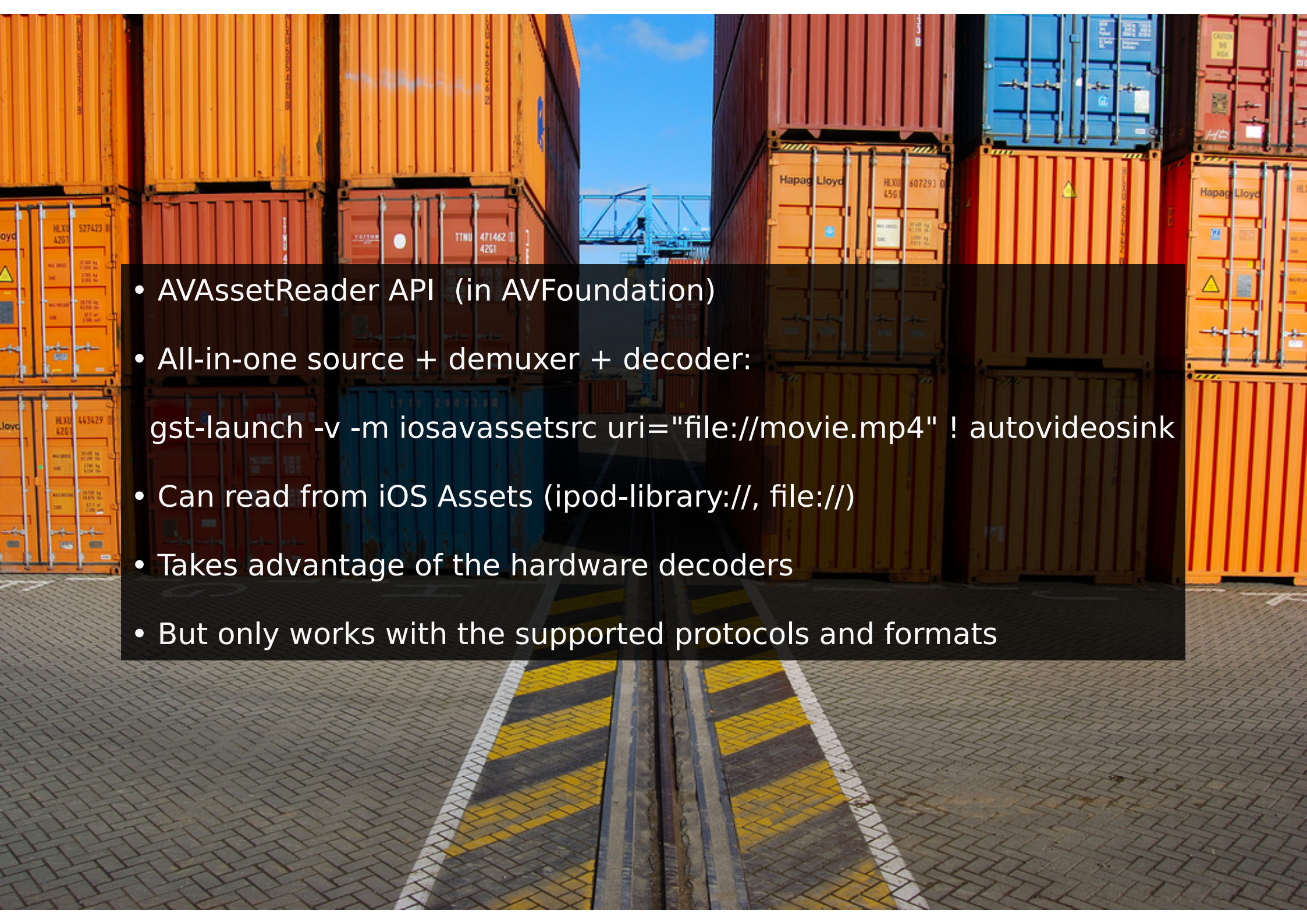


AVFoundation video source

- Output formats: NV12, UYVY, YUY2
- Output resolutions: 192x144 352x288 480x360 640x480 1280x720 1980x1080
- Supports device selection



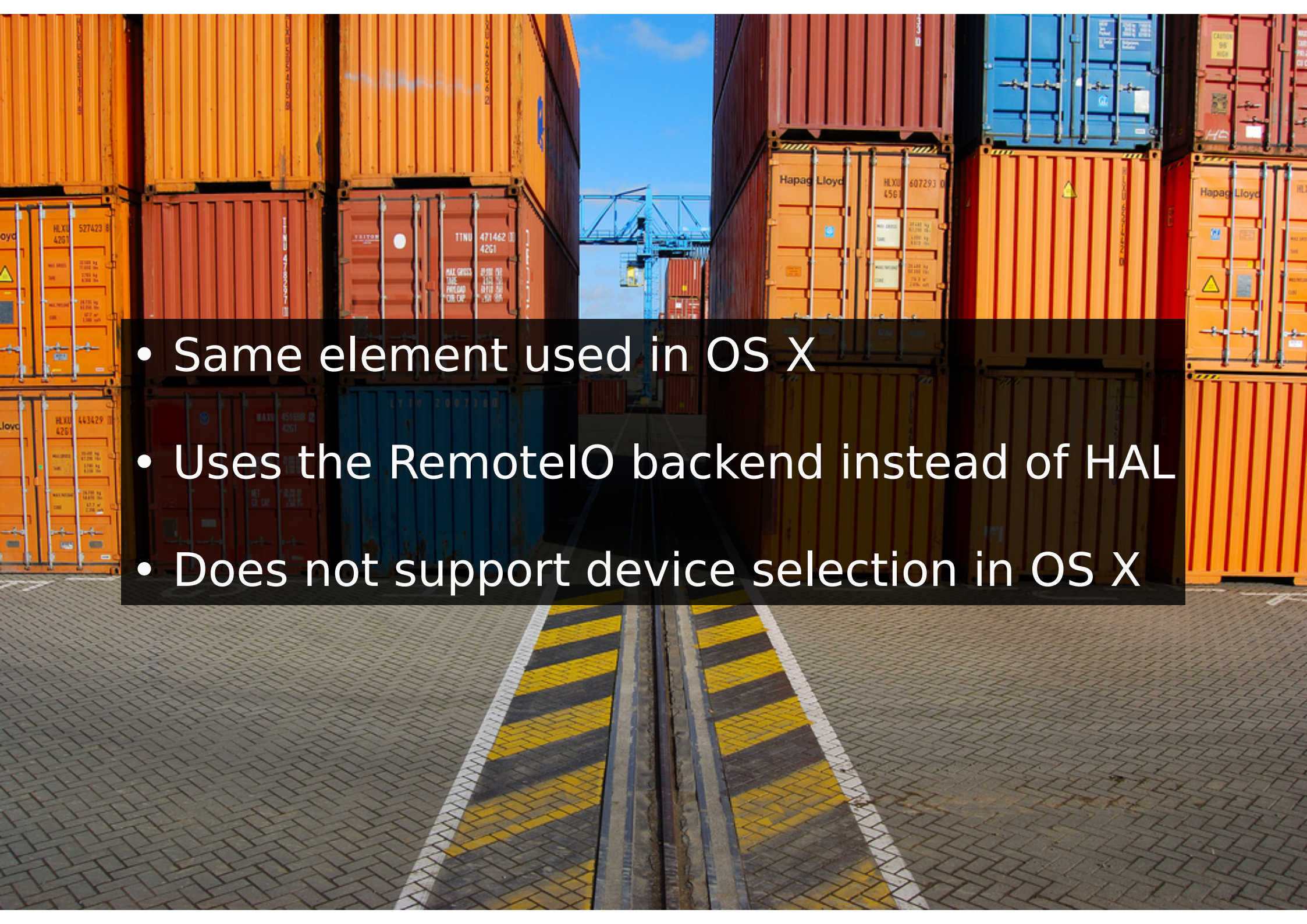
AVAssetReader URI decoder iosavassetsrc


- 
- AVAssetReader API (in AVFoundation)
 - All-in-one source + demuxer + decoder:
`gst-launch -v -m iosavassetsrc uri="file://movie.mp4" ! autovideosink`
 - Can read from iOS Assets (ipod-library://, file://)
 - Takes advantage of the hardware decoders
 - But only works with the supported protocols and formats



CoreAudio

osxaudiosrc

- 
- Same element used in OS X
 - Uses the RemoteIO backend instead of HAL
 - Does not support device selection in OS X



3. Building GStreamer with cerbero

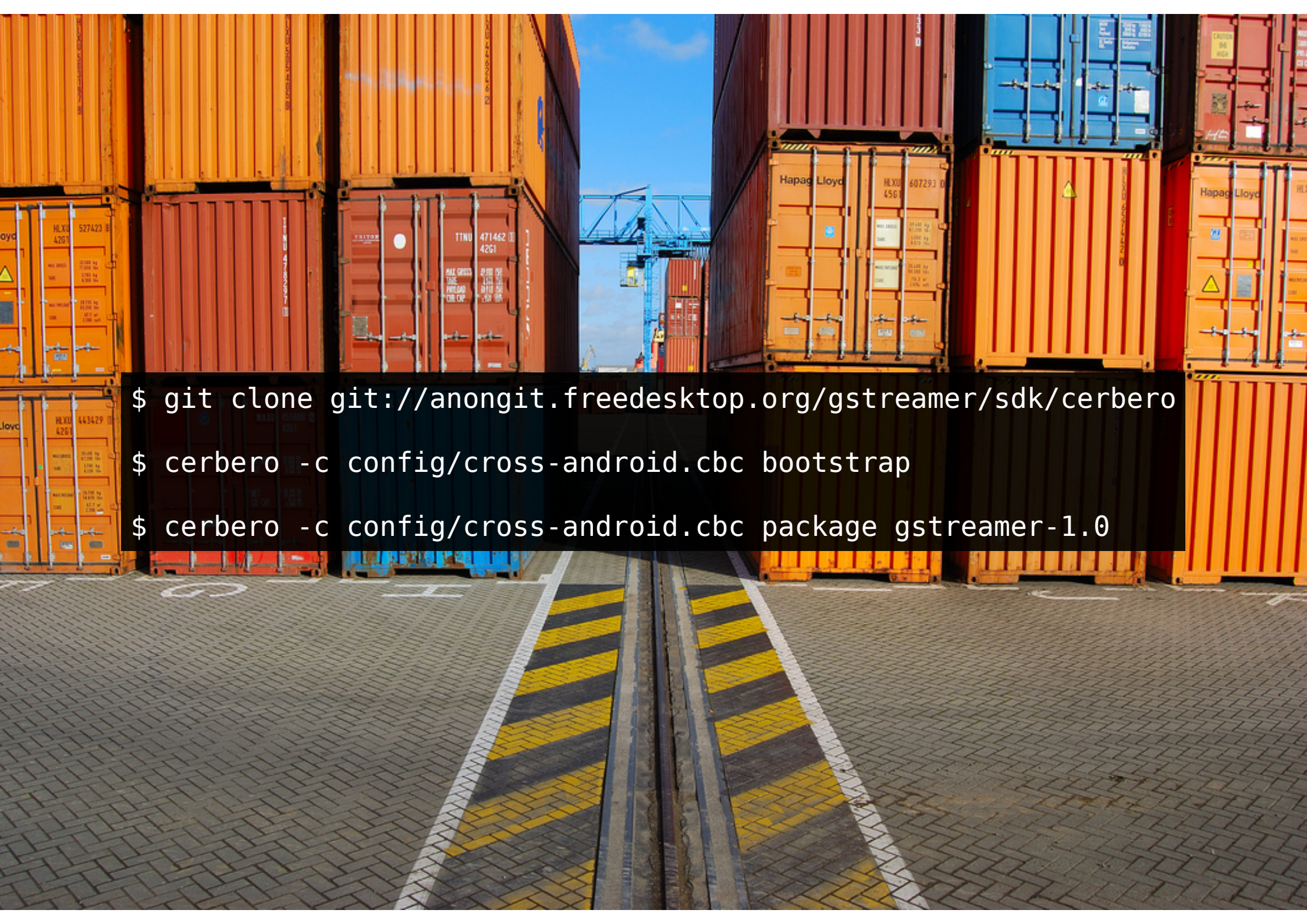
Cerbero

- We use a build system named Cerbero: 3-headed dog which guards the doors of the Underworld (Linux, Windows, OS X)
- Same build system used in all platforms.
- Re-use of upstream packaging system.
- Native packaging: Windows .msi, OS X .pkg, RPM and DEB
- Easy to maintain
- Easy to add new packages or 3rd party plugins



Cerberero

- Bootstrapping facilities
- Platform definition files:
very easy to add support for new platforms/toolchains
- Build config files:
configures build for different uses



```
$ git clone git://anongit.freedesktop.org/gstreamer/sdk/cerbero  
$ cerbero -c config/cross-android.cbc bootstrap  
$ cerbero -c config/cross-android.cbc package gstreamer-1.0
```





Available configuration files:

Android

- cross-android.cbc: Android armv5
- cross-android-armv7.cbc: Android armv7
- cross-android-x86.cbc: Android armv7

iOS

- cross-ios-x86.cbc: iOS x86 (emulator)
- cross-ios-armv7.cbc: iOS armv7 (devices)
- cross-ios-universal.cbc: iOS armv7+x86



4. Integration with the build infrastructure (embedding GStreamer)



Android

- Static plugins
- Static linking with re-locatable archives.
- A single shared library with everything:
`libgstreamer_android.so`
- Integration with ndk-build to link this shared library:
 - Complies with the LGPL requirement.
 - Allows selecting only the plugins being used.

Android GStreamer application

Java Application

Java Native Interface

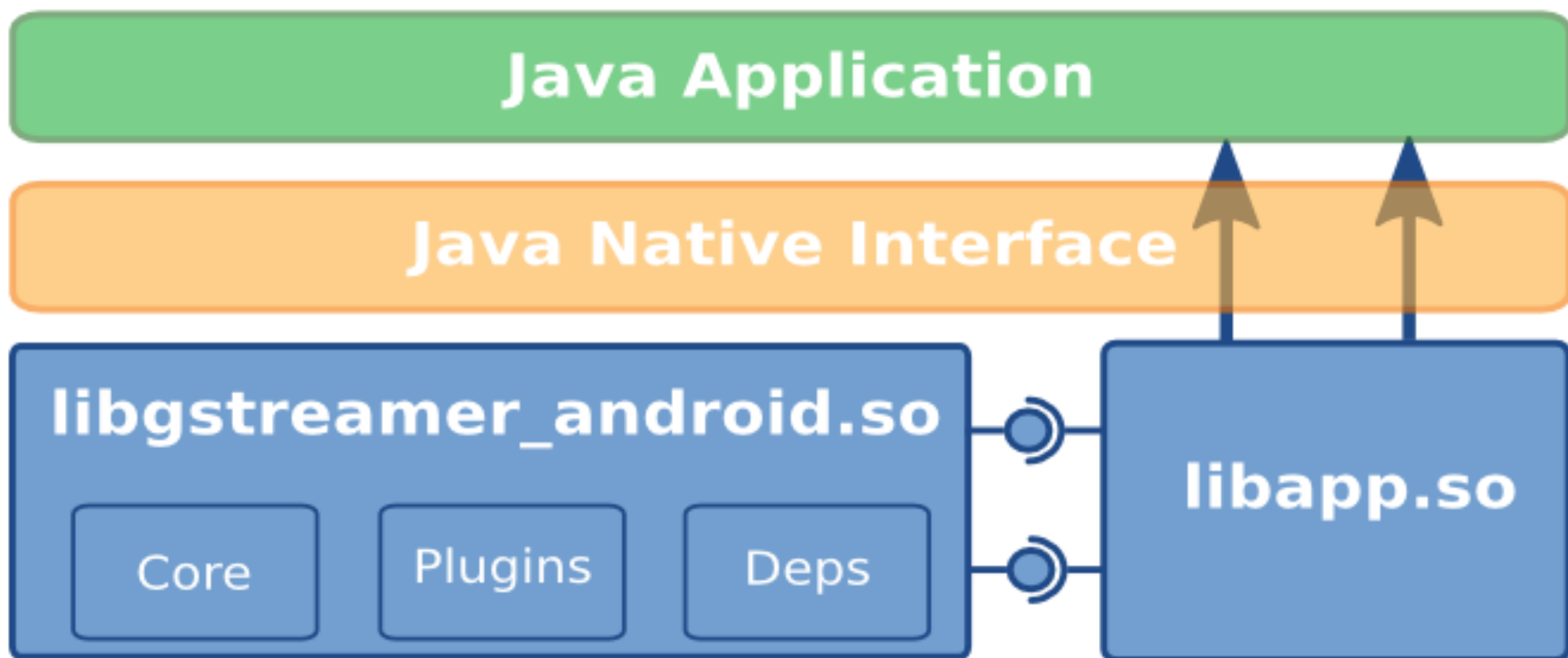
libgstreamer_android.so

Core

Plugins

Deps

libapp.so





Android integration with the NDK



```
LOCAL_PATH := $(call my-dir)
```

```
include $(CLEAR_VARS)
```

```
LOCAL_MODULE := hello-jni
```

```
LOCAL_SRC_FILES := hello-jni.c
```

```
include $(BUILD_SHARED_LIBRARY)
```

```
include $(CLEAR_VARS)
```




```
LOCAL_PATH := $(call my-dir)
```

```
include $(CLEAR_VARS)
```

```
LOCAL_MODULE := hello-jni
```

```
LOCAL_SRC_FILES := hello-jni.c
```

```
LOCAL_SHARED_LIBRARIES := gstreamer_android
```

```
include $(BUILD_SHARED_LIBRARY)
```

```
include $(CLEAR_VARS)
```

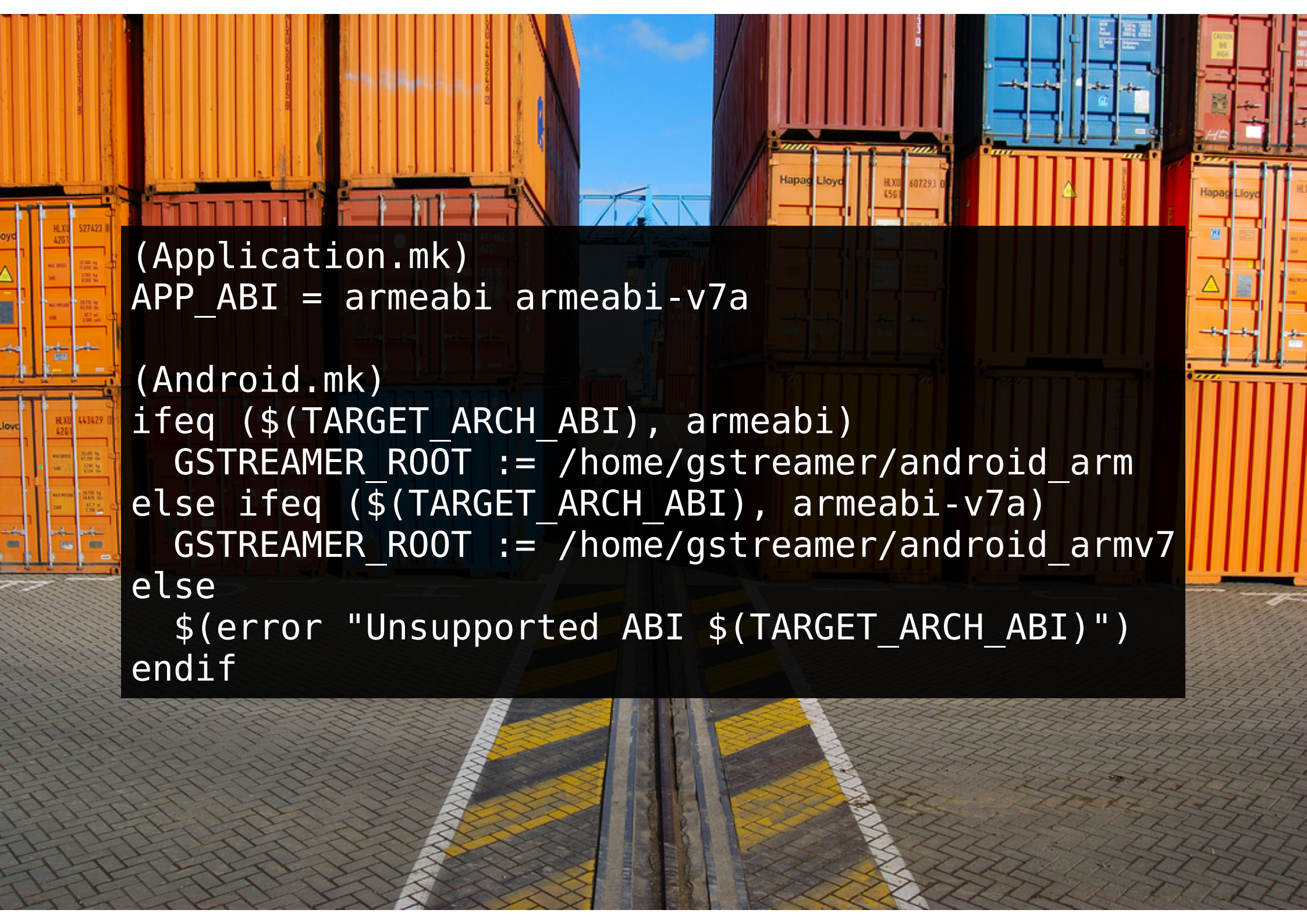
```
include $(GSTREAMER_NDK_BUILD_PATH)/plugins.mk
```

```
GSTREAMER_ROOT := /home/cerbero/android_arm
```

```
GSTREAMER_PLUGINS = $(GSTREAMER_PLUGINS_CORE)  
                    $(GSTREAMER_PLUGINS_CODECS)
```

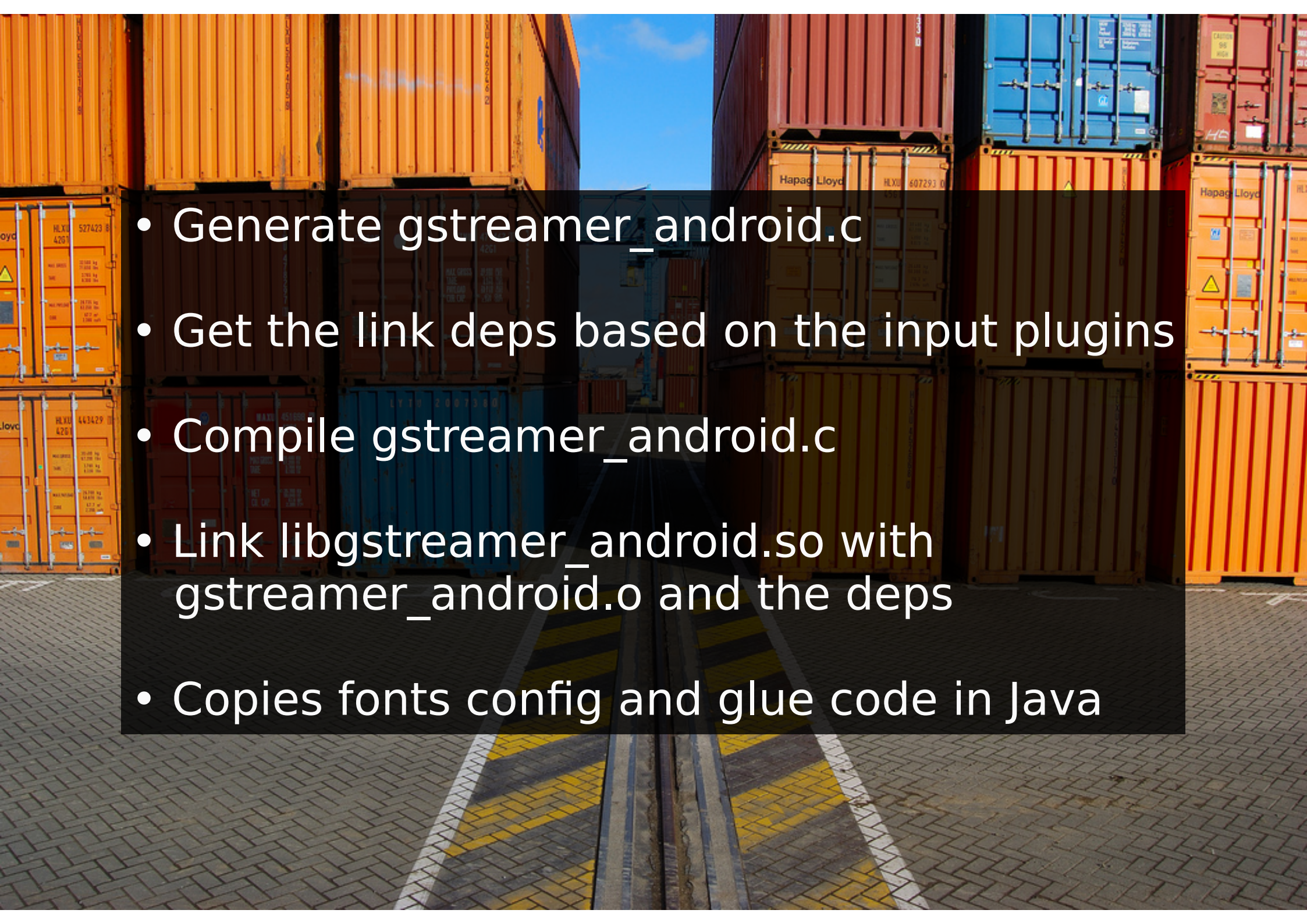
```
GSTREAMER_EXTRA_DEPS := json-glib-1.0
```

```
include $(GSTREAMER_NDK_BUILD_PATH)/gstreamer.mk
```

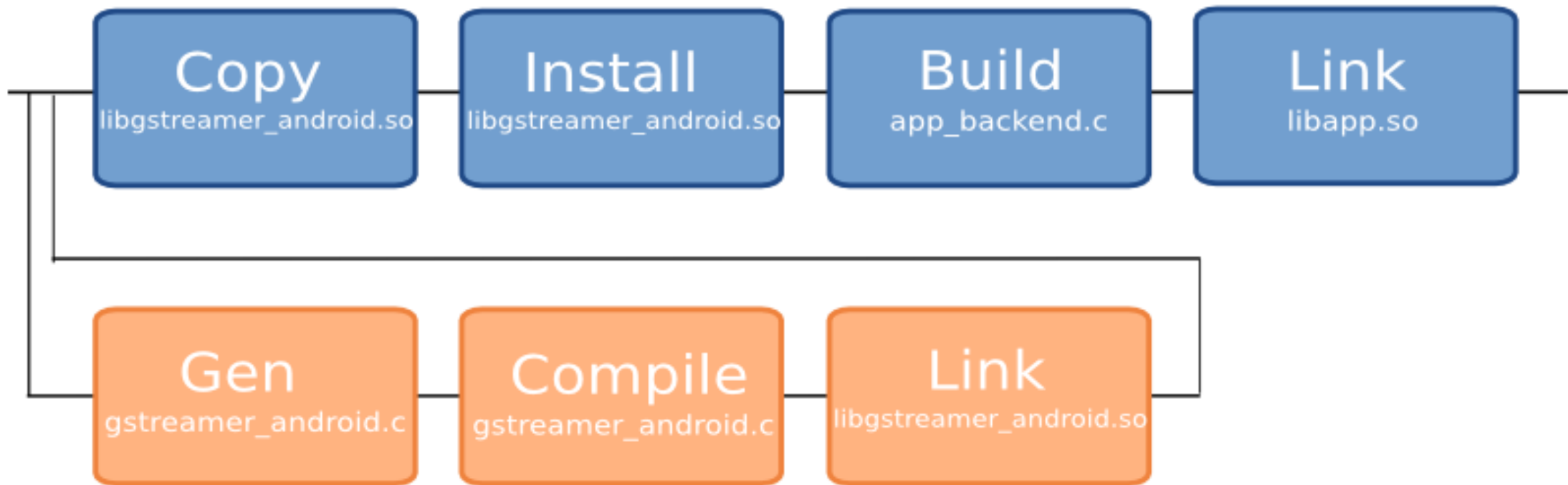




```
(Application.mk)
APP_ABI = armeabi armeabi-v7a
```

```
(Android.mk)
ifeq ($(TARGET_ARCH_ABI), armeabi)
    GSTREAMER_ROOT := /home/gstreamer/android_arm
else ifeq ($(TARGET_ARCH_ABI), armeabi-v7a)
    GSTREAMER_ROOT := /home/gstreamer/android_armv7
else
    $(error "Unsupported ABI $(TARGET_ARCH_ABI)")
endif
```


- 
- Generate `gststreamer_android.c`
 - Get the link deps based on the input plugins
 - Compile `gststreamer_android.c`
 - Link `libgststreamer_android.so` with `gststreamer_android.o` and the deps
 - Copies fonts config and glue code in Java

ndk-build build steps



- 
- We use libtool libraries to resolve link deps
 - Libtool can't be used for portability issues
 - A small libtool replacement in GNU Makefile + sed
 - Portable (works on Windows too)
 - Supports relocations of .la files
 - Much faster than libtool



iOS

- Static linking
- iOS Framework to work with XCode
- Big archive with all the static libraries
- Selection of plugins with `#ifdef`'s registering static plugins
- The strip step takes care of removing the extra archives and symbols that are not used by the application



(gst_ios_init.h)

```
#define GST_IOS_PLUGINS_CORE
//#define GST_IOS_PLUGINS_CAPTURE
//#define GST_IOS_PLUGINS_CODECS_RESTRICTED
//#define GST_IOS_PLUGINS_ENCODING
//#define GST_IOS_PLUGINS_DVD
//#define GST_IOS_PLUGINS_CODECS_GPL
//#define GST_IOS_PLUGINS_NET_RESTRICTED
//#define GST_IOS_PLUGINS_SYS
//#define GST_IOS_PLUGINS_VIS
#define GST_IOS_PLUGINS_PLAYBACK
#define GST_IOS_PLUGINS_EFFECTS
#define GST_IOS_PLUGINS_CODECS
#define GST_IOS_PLUGINS_NET
```



```
#include "gst_ios_init.h"


#if defined(GST_IOS_PLUGIN_COREELEMENTS) || defined(GST_IOS_PLUGINS_CORE)
    GST_PLUGIN_STATIC_DECLARE(coreelements);
#endif

void
gst_ios_init (void)
{
    GstPluginFeature *plugin;
    GstRegistry *reg;
    NSString *resources = [[NSBundle mainBundle] resourcePath];
    NSString *tmp = NSTemporaryDirectory();

    const gchar *tmp_dir = [tmp UTF8String];
    [...]
    g_setenv ("TMP", tmp_dir, TRUE);
    [...]

    gst_init (NULL, NULL);
    #if defined(GST_IOS_PLUGIN_COREELEMENTS) || defined(GST_IOS_PLUGINS_CORE)
        GST_PLUGIN_STATIC_REGISTER(coreelements);
    #endif

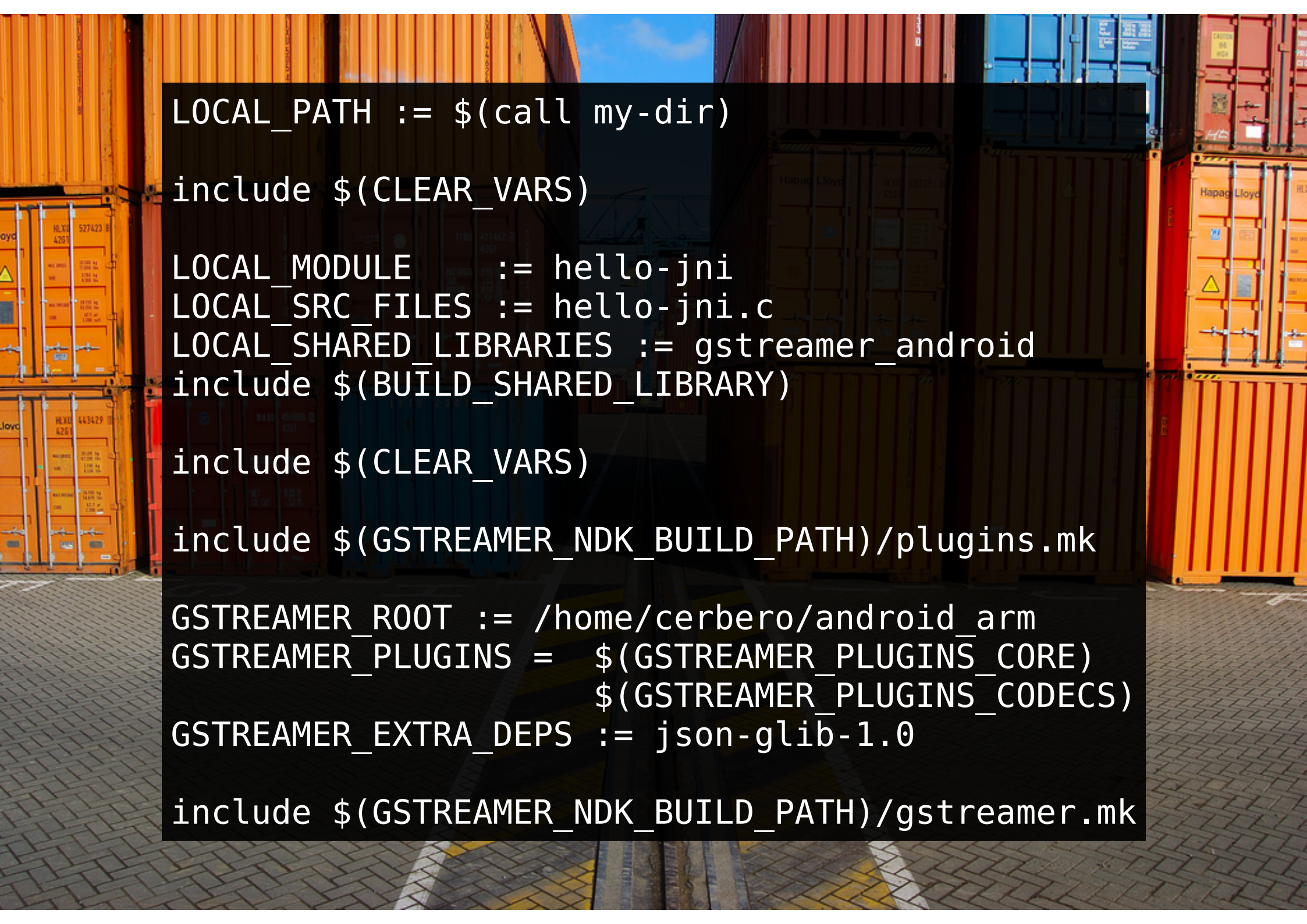
    /* Lower the ranks of filesrc and giosrc so iosavassetsrc is
     * tried first in gst_element_make_from_uri() for file:// */
    reg = gst_registry_get_default();
    plugin = gst_registry_lookup_feature(reg, "filesrc");
    if (plugin)
        gst_plugin_feature_set_rank(plugin, GST_RANK_SECONDARY);
}
```

5. Installing GStreamer and creating your first application

Android

- Distributed with a tarball
- Includes:
 - Headers
 - Static libraries (.a)
 - Libtool libraries (.la)
 - NDK integration makefiles
- Only requires to extract the tarball
- Create a project in Eclipse and add native support project→Android Tools→Add Native Support
- Finally edit your Android.mk with the path to the root



```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE := hello-jni
LOCAL_SRC_FILES := hello-jni.c
LOCAL_SHARED_LIBRARIES := gstreamer_android
include $(BUILD_SHARED_LIBRARY)

include $(CLEAR_VARS)

include $(GSTREAMER_NDK_BUILD_PATH)/plugins.mk

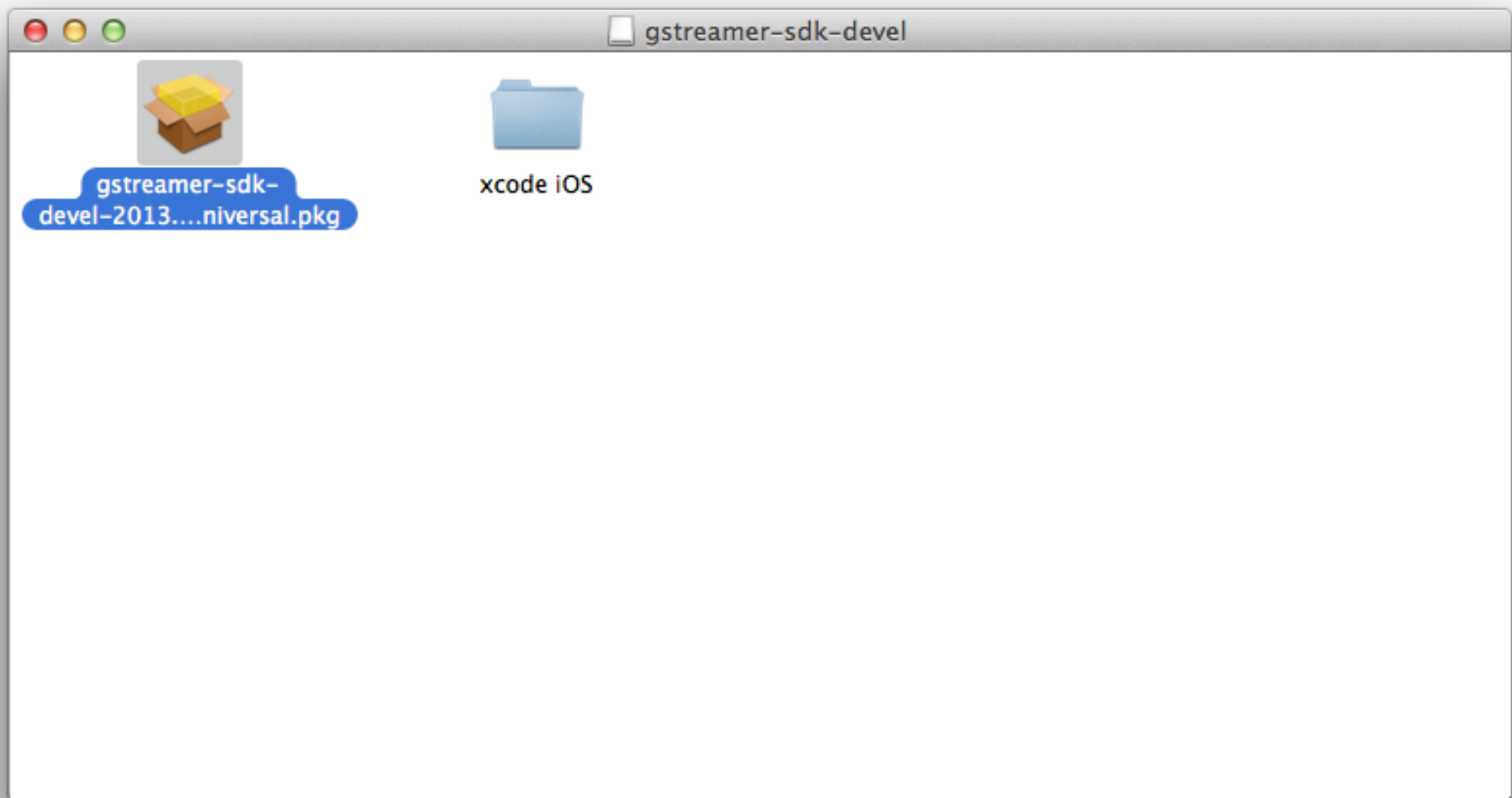
GSTREAMER_ROOT := /home/cerbero/android_arm
GSTREAMER_PLUGINS = $(GSTREAMER_PLUGINS_CORE)
                    $(GSTREAMER_PLUGINS_CODECS)
GSTREAMER_EXTRA_DEPS := json-glib-1.0

include $(GSTREAMER_NDK_BUILD_PATH)/gstreamer.mk
```



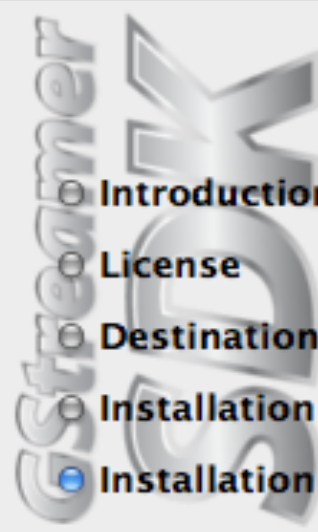

ios

- Distributed as DMG image file
- Contains a .pkg to install the framework and a folder with the tutorials
- The .pkg installs the Framework and the XCode templates





Install GStreamer SDK (Development Files)



- ☐ Introduction
- ☐ License
- ☐ Destination Select
- ☐ Installation Type
- ☒ Installation
- ☐ Summary

Installing GStreamer SDK (Development Files)

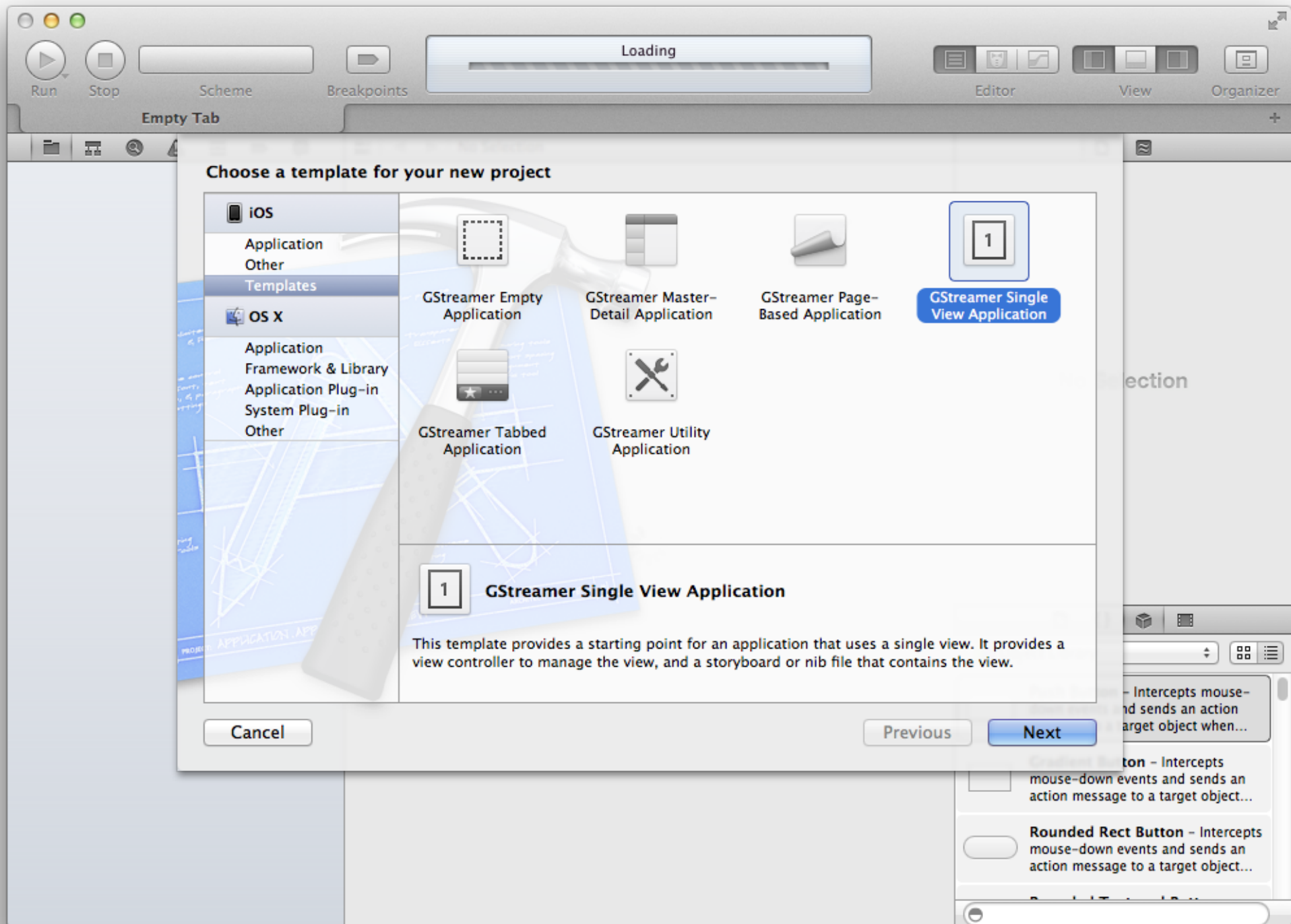
The software was successfully installed.

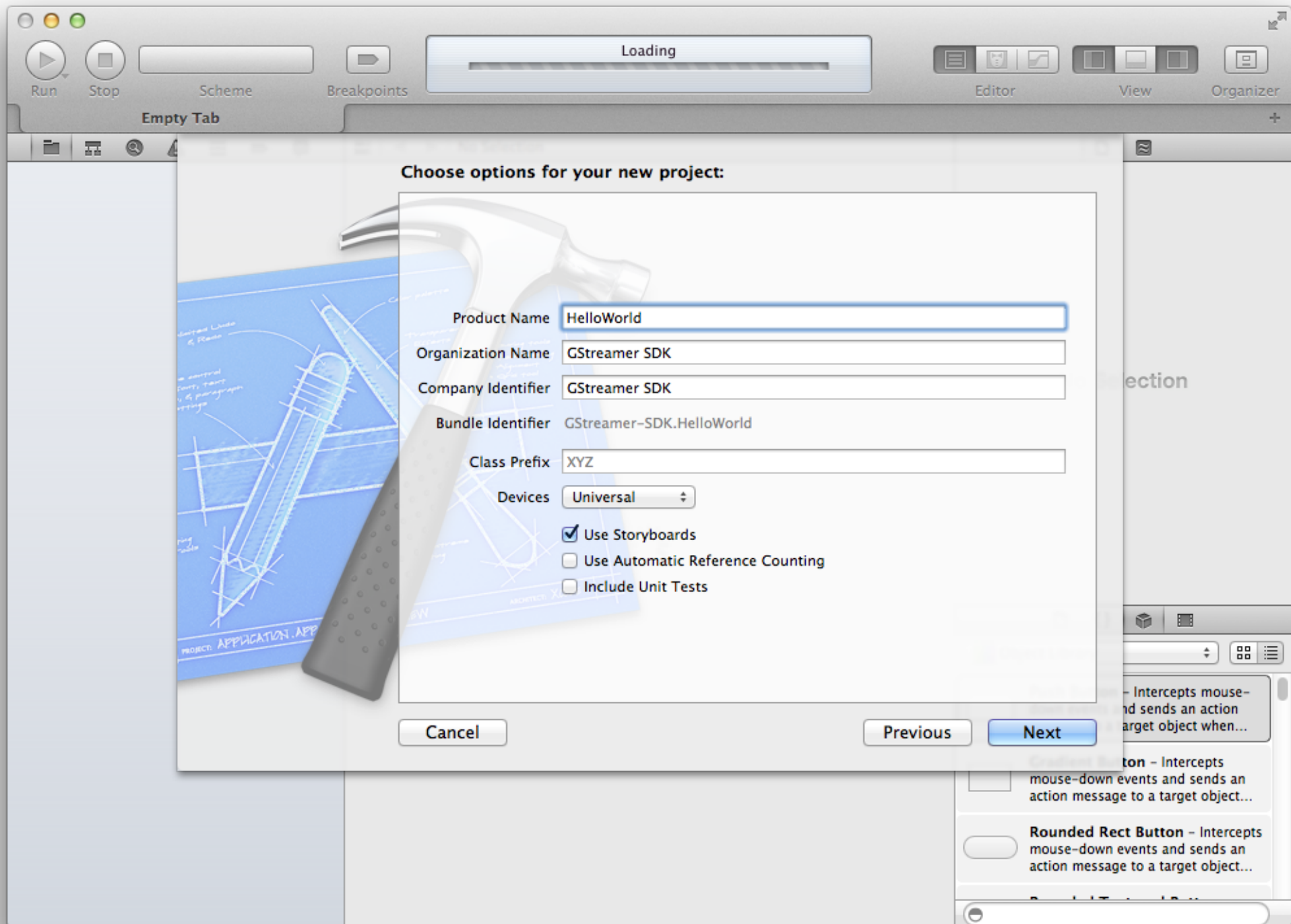


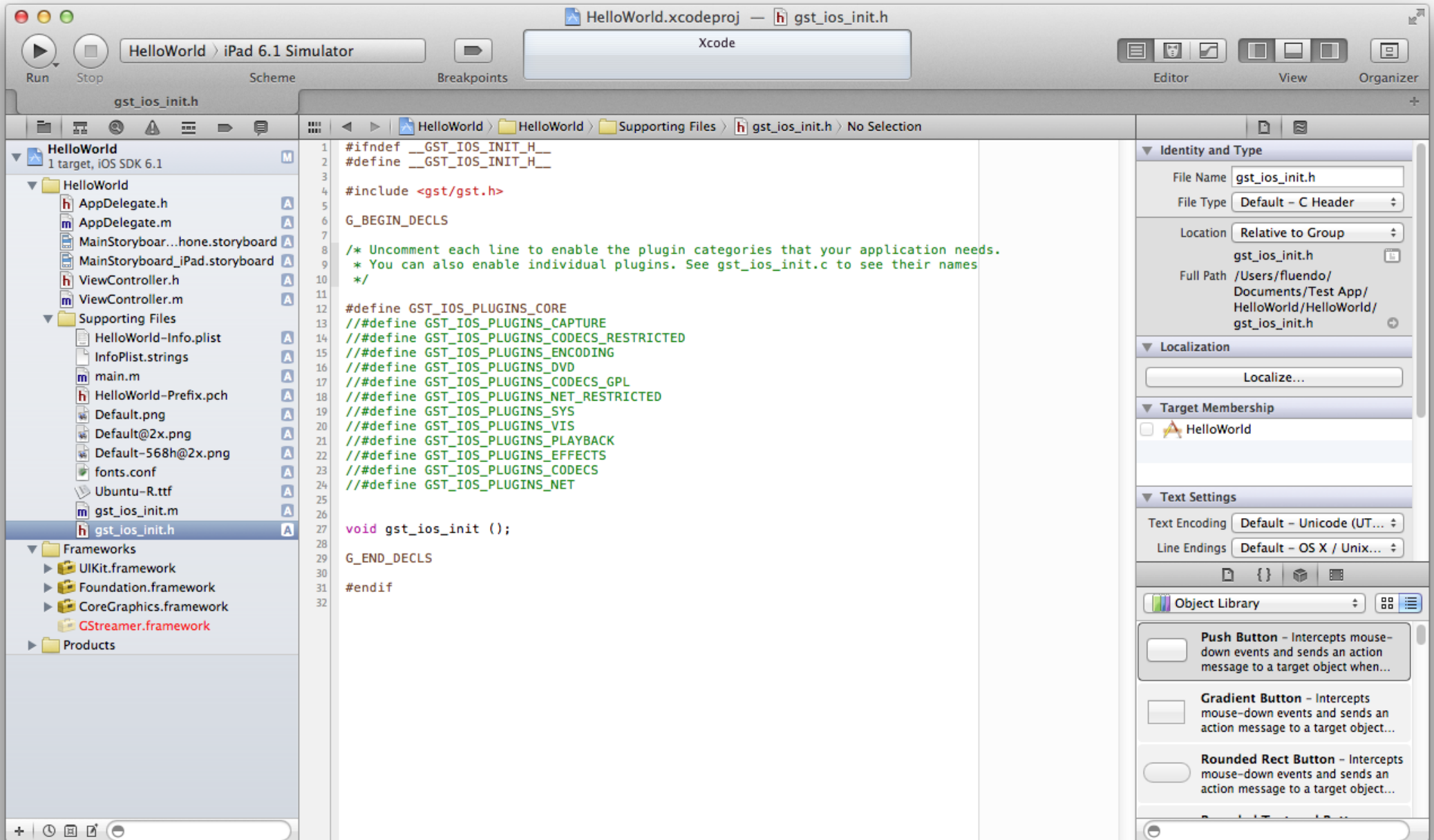
Install time remaining: About 2 minutes

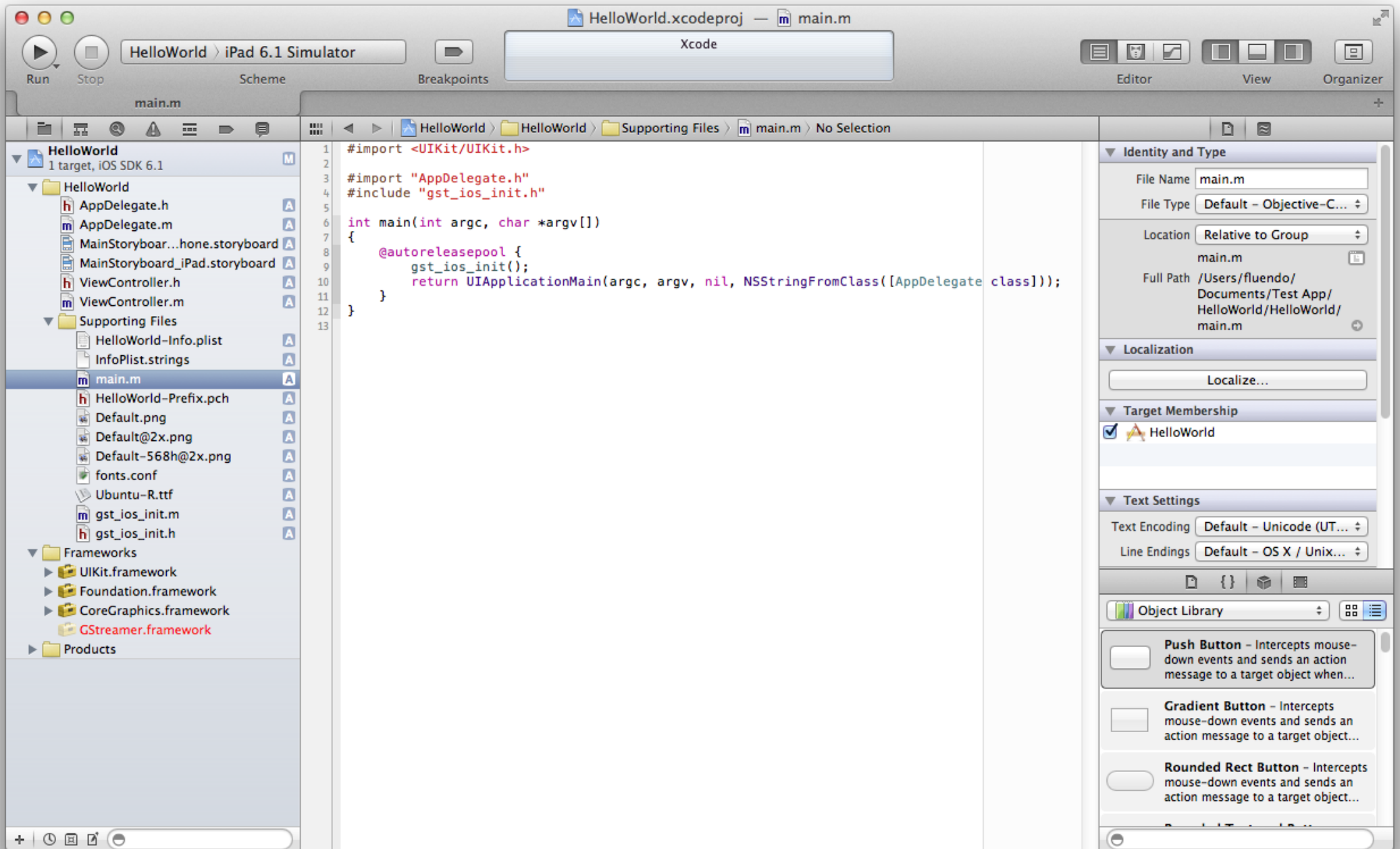
Go Back

Continue









THANK YOU

¿Questions?





Thanks!